

Prediction of Aerodynamic Coefficients using Neural Networks for Sparse Data

T. Rajkumar
Computer Scientist, SAIC, NASA
and

Jorge Bardina
Computer Scientist, NASA

Mail Stop 269-2 Moffett Field, California, USA 94035
rajkumar, jbardina@mail.arc.nasa.gov

Abstract

Basic aerodynamic coefficients are modeled as functions of angles of attack and sideslip with vehicle lateral symmetry and compressibility effects. Most of the aerodynamic parameters can be well-fitted using polynomial functions. In this paper a fast, reliable way of predicting aerodynamic coefficients is produced using a neural network. The training data for the neural network is derived from wind tunnel test and numerical simulations. The coefficients of lift, drag, pitching moment are expressed as a function of alpha (angle of attack) and Mach number. The results produced from preliminary neural network analysis are very good

Introduction

Wind tunnels use scaled models to characterize aerodynamic coefficients. The wind tunnel data, in original form, are unsuitable for use in piloted simulations because data obtained in different wind tunnels with different scale models of the same vehicle are not always consistent. Fitting a smooth function through the wind tunnel data results in smooth derivatives of the data. The smooth derivatives are important in performing stability analyses. Traditionally, the approach considered to describe the aerodynamics of the vehicle included developing, wherever possible, a polynomial description of each aerodynamic function (B. Jackson and C. Cruz 1992). This ensured a smooth continuous function and removed some of the scatter in the wind tunnel data. Also, measurements of the same coefficient from two different wind tunnels are usually taken at dissimilar values of angle of attack and sideslip, and some means of reconciling the two dissimilar sets of raw data were needed. This curve fitting procedure is unnecessary for few coefficients. The curve fitting method used to generate the parameters for each polynomial description is an unweighted least squares algorithm. For the most part, the polynomial equations are generated using sparse data from wind tunnel experiments.

Due to sparcity of data, mostly it will be defined as a linear type of function. When more data are available, flight control system designs will need to be revisited to allow for minor nonlinearities in control effects.

Wind tunnel testing can be slow and costly due to high personnel overhead and intensive power utilization. Although manual curve fitting can be done, it is highly efficient to use a neural network (Magnus Norgaard, Jorgensen C and James Ross 1997, M.M.Rai and N.K.Madavan 2000 and Ching F.Lo, J.L.Zhao and DeLoach R 2000) to define complex relationship between variables. This paper is organized as follows: A short introduction to neural network and learning followed by a section that will introduce the data set. Then results will be discussed to find an optimal solution to the various aerodynamic coefficients. The final section will conclude optimizing the neural network and research directions.

Neural Network

A neural network is conceptually comprised of a collection of nodes and connections (Rumelhart, Hinton and Williams 1986; Wasserman 1989; Simpson 1990; Lau 1992; Masters 1993; Jondarr 1996; Kartalopoulos 1996). The basic elements of a network are called neurons; they represent the sites that process information. The interconnecting links between the processing units, or neurons, are called synapses. Each synapse can be characterized by a weight, which is represented by a numerical value. All neurons in the adjacent layers are connected and the flow of information is restricted to the forward direction. The network consists of three layers: input layer, hidden layer(s) and output layer. The hidden layer enables the network to learn relationships between input-output variables through suitable mappings. Among the many neural network models, the backpropagation algorithm is one of the better known and frequently used.

Back propagation (Rumelhart, Hinton and Williams 1986) was created by generalizing the Widrow-Hoff learning rule (Widrow and Hoff 1960; Widrow and Stearn 1985; Widrow et al 1987) to multiple-layer networks and nonlinear differentiable transfer functions. Input - output pairs are used to train a network until it can approximate a function. Back propagation was the first practical method for training a multiple layer feed forward network. A neural network's initial weights are simply random numbers, which change during training. Training consists of presenting actual experimental data to the neural network and using a mathematical algorithm—the back propagation algorithm—to adjust the weights. Each pair of patterns goes through two stages of activation: a forward pass and a backward pass. The forward pass involves presenting a sample input to the network and letting the activations flow until they reach the output layer. During the backward pass, the network's actual output (from the forward pass) is compared with the target output and errors are computed for the output units. Adjustments of weights are based on the difference between the correct and computed outputs. Once each observation's computed and actual outputs are within the specified error tolerance, training stops and the neural network is ready for use: given a new input observation, it will estimate what the corresponding output values should be. After extensive training, the network eventually establishes the input-output relationships through the adjusted weights on the network.

Learning

Learning in neural network is typically accomplished using examples. This is also called "training" of the neural network because the learning is achieved by adjusting the connection weights¹ in the neural network iteratively so that trained (or learned) neural network can perform certain tasks. Learning in neural network can roughly be divided into supervised, unsupervised, and reinforcement learning. Supervised learning is based on direct comparison between the actual output of a neural network output and the desired correct output, also known as the target output. It is often formulated as the minimization of an error function such as the total mean square error between the actual output and the desired output summed over all available data. A gradient descent-based optimization algorithm such as backpropagation can then be used to adjust connection weights in the neural network iteratively in order to minimize the error. Reinforcement learning is a special case of supervised learning where the exact desired output is unknown. It is based only on the information of whether or not the actual output is correct. Unsupervised learning

is solely based on the correlations among input data. No information on "correct output" is available for learning. The essence of a learning algorithm is the learning rule, i.e., a weight-updating rule which determines how connection weights are changed. Examples of popular learning rules include the delta rule, the Hebbian rule, the anti-Hebbian rule, and the competitive learning rule. More detailed discussion of neural network and learning algorithms are beyond the scope of this paper. The success of the neural network depends greatly on defining the influencing parameters for the problem, definition of suitable neural network architecture and the learning algorithm.

Data Set for Aerodynamic Models

Aerodynamic control systems can be divided into two categories viz., control surfaces and aerodynamics controls. In this paper, aerodynamic controls and models are the focus. The variables involved in aerodynamic controls are angle of attack (α), sideslip angle (β), elevon deflections (δ_e), aileron deflections (δ_a), rudder deflection (δ_R), speed brake deflection (δ_{SB}), landing gear effects and ground effects. The general equations of forces (lb) and moments (ft-lb) for key parameters are listed in the following tables 1 and 2 (B. Jackson and C. Cruz 1992).

Forces (lb)	Model
Lift	$L=CL.q.S$
Drag	$D=CD.q.S$
Side-force	$FY=CY.q.S$

Table 1. Aerodynamic forces

Moments (ft-lb)	Model
Pitching	$PM=Cm.q.S.c+(L.\cos\alpha+D.\sin\alpha).X_{MRC}+(L.\sin\alpha-D.\cos\alpha).Z_{MRC}$
Rolling	$RM=Cl.q.S.b+FY.Z_{MRC}$
Yawing	$Cn.q.S.b+FY.X_{MRC}$

Table 2 Aerodynamic Moments

The aerodynamic coefficients involved in the above equations are presented.

Longitudinal aerodynamic coefficients

Lift Coefficient CL:

$$CL = CL_{BAS}(\alpha, M) + \Delta CL_{\delta FLAPS}(\delta F) \cdot \delta F + \Delta CL_{SPEEDBRAKE}(\alpha, \delta_{SB}) + \Delta CL_{LG}(\delta LG) + \Delta CL_{gc}(h/b) + \Delta CL_{q,q}(\alpha, M) \cdot q \cdot c/2U + \Delta CL_{\alpha}(\alpha, M) \cdot \alpha \cdot c/2U$$

Drag Coefficient CD:

¹ Thresholds (biases) can be viewed as connection weights with fixed input 1.

$$CD = CD_{BAS}(\alpha, M) + \Delta CD_{\delta FLAPS}(\delta F) \cdot \delta F + \Delta CD_{SPEEDBRAKE}(\alpha, \delta SB) + \Delta CD_{LG}(\delta LG) + \Delta CD_{ge}(h/b) + \Delta CD_{q(\alpha)} \cdot q \cdot c/2U$$

Pitching Moment Coefficient C_m :

$$C_m = C_{m_{BAS}}(\alpha, M) + \Delta C_{m_{\delta FLAPS}}(\delta F) \cdot \delta F + \Delta C_{m_{SPEEDBRAKE}}(\alpha, \delta SB) + \Delta C_{m_{LG}}(\delta LG) + \Delta C_{m_{ge}}(h/b) + \Delta C_{m_{q(\alpha, M)}} \cdot q \cdot c/2U + \Delta C_{m_{\alpha}}(\alpha, M) \cdot \alpha \cdot c/2U$$

Lateral aerodynamic coefficients

Side force coefficient C_Y :

$$C_Y = C_{Y_{SB}}(\alpha, M) \cdot \beta + \Delta C_{Y_{\delta RUDDER}}(\delta R) \cdot \delta R + \Delta C_{Y_{\delta AILERON}}(\delta A) \delta A + \Delta C_{Y_{LG\Delta\beta}}(\delta LG) \beta + \Delta C_{Y_{GE\Delta\beta}}(h/b) \beta + \Delta C_{Y_p}(\alpha) \cdot p \cdot b/2U + \Delta C_{Y_r}(\alpha) \cdot r \cdot b/2U$$

Rolling Moment Coefficient C_l :

$$C_l = C_{l_{SB}}(\alpha, M) \cdot \beta + \Delta C_{l_{\delta RUDDER}}(\delta R) \cdot \delta R + \Delta C_{l_{\delta AILERON}}(\delta A) \delta A + \Delta C_{l_{LG\Delta\beta}}(\delta LG) \beta + \Delta C_{l_{GE\Delta\beta}}(h/b) \beta + \Delta C_{l_p}(\alpha) \cdot p \cdot b/2U + \Delta C_{l_r}(\alpha) \cdot r \cdot b/2U$$

Yawing Moment Coefficient C_n :

$$C_n = C_{n_{SB}}(\alpha, M) \cdot \beta + \Delta C_{n_{\delta RUDDER}}(\delta R) \cdot \delta R + \Delta C_{n_{\delta AILERON}}(\delta A) \delta A + \Delta C_{n_{LG\Delta\beta}}(\delta LG) \beta + \Delta C_{n_{GE\Delta\beta}}(h/b) \beta + \Delta C_{n_p}(\alpha) \cdot p \cdot b/2U + \Delta C_{n_r}(\alpha) \cdot r \cdot b/2U$$

Above equations depend basically on angle of attack and Mach number with little increments of other factors. The above equation can be expressed as a function of angle of attack and Mach number and it resembles a simple polynomial expression.

Depending on the geometry and mass properties of the vehicle, aerodynamics coefficients will vary. The general parameters are tabulated in table 3.

Parameters	Ranges of values
Angle of attack (degrees)	$-10 < \alpha < 50$
Side angle (degrees)	$-20 < \beta < 20$
Mach number	$M \leq 0.9$
Surface deflection (degrees)	$-15 < \delta_{levons}(\text{flaps}) < 15$ $-20 < \delta_{rudder} < 20$ $-20 < \delta_{aileron} < 20$ $0 < \delta_{speedbrake} < 80$

Table 3 Range of values involved in aerodynamic coefficients

If a regression analysis can fit a data set very well, then neural network can perform much better than regression techniques. Inputs considered for determining base

coefficients are angle of attack and Mach number. The output of the neural network is the coefficients of aerodynamic model. As a good training data set for a particular vehicle type, geometry and mass are selected from any wind tunnel test. Some times if the data set is not available from experiments for wind tunnels, a good training data set can be derived from numerical computations from Euler or Navier stokes or Vortex lattice method. This data set consists of a comprehensive input and output tuple for an entire parameter space. Once training data set is defined, sparse data collected from experiments can be interpolated and extended for the entire range of data using a trained neural network. This will avoid repeating the entire experiments in the wind tunnel. Once training data set is selected, one must determine the type of neural network architecture and transfer functions that will be used to interpolate the sparse data. The next section will discuss the selection procedure of the neural network architecture and transfer functions used in this work.

Neural Network Architecture

In this paper, interpolating for coefficient of lift is discussed for sparse data set. The rest of the various aerodynamic coefficients will be repeated with the same architecture of neural network with respect to corresponding data set. The problem of defining neural network architectures (Freeman J. A and Skapura D.M 1992, Hagan M.T., Demuth H.B and Beale 1996) can be divided into the following categories: (i) type of neural network (whether three layer or four layer, etc.); (ii) number of hidden neurons; (iii) type of transfer functions (Elliott D.L 1993); (iv) training algorithm; and (v) over and under fitting of the results and validation of neural network output. If the function consists of a finite number of points, a three layer neural network is capable of learning the function. Since the availability of data is limited, the type of neural network considered for this problem is a three layer neural network with input layer, hidden layer and output layer. The input layer will have two input neurons (alpha and Mach number) and output layer will represent one neuron (coefficient of lift). Domain data has specific definite bounds, rather than having no limits. The number of hidden neurons is defined based on the efficient fitting of the data.

For determining an appropriate (hopefully optimal or near-optimal) number of hidden units (Lawrence S, Lee Giles and Ah Chung Tsoi 1996), we construct a sequence of networks with increasing number of hidden neurons from 2 to 20. More than 20 hidden neurons cause an over fitting of the results (Lawrence S, Lee Giles and Ah chung Tsoi 1997). Each neuron in the network is fully connected and uses all available input variables. First, a network with a small number of hidden units is trained using random

initial weights. Iteratively, a larger network is constructed (up to the 20 hidden neurons) and the network results are compared with the expected results. Activation functions also play a key role in producing the best network results. The transfer function is a nonlinear function that when applied to the net input of a neuron, determines the output of the neuron. The majority of neural networks use a sigmoid function (S-shaped). A sigmoid function is defined as a continuous real-valued function whose domain is the Reals, whose derivative is always positive, and whose range is bounded. In this aerodynamic problem, only a sigmoid function can alone produce an efficient fit. To get a best fitting, it is suggested to use different kinds of transfer functions for different layers of network. However, functions such as "tanh" that produce both positive and negative values tend to yield faster training than functions that produce only positive values such as sigmoid, because of better numerical conditioning. Numerical condition affects the speed and accuracy of most numerical algorithms. Numerical condition is especially important in the study of neural networks because ill-conditioning is a common cause of slow and inaccurate results from backprop-type algorithms. Activation functions for the hidden units are needed to introduce nonlinearity into the network. Without nonlinearity, hidden units would not make nets more powerful than just plain perceptrons (which do not have any hidden units, just input and output units). The reason is that a linear function of linear functions is again a linear function. However, it is the nonlinearity (i.e., the capability to represent nonlinear functions) that makes multilayer networks so powerful. Three types of activation functions are used in neural networks namely linear, sigmoid and hyperbolic tangent.

The training epoch is restricted to 1000 cycles {present a data set, measure error, update weights}. The learning rate and momentum is selected appropriately to get faster convergence of the network. The input and output values are scaled to range [0.1, 0.9] to ensure that the output will lie in the output region of the nonlinear sigmoid transfer function. Presentable variable values lie in between 0.1 and 0.9 (0.1 and 0.9 inclusive). The scaling is performed using the following equation

$$A = r(V - V_{\min}) + A_{\min}$$

$$r = \frac{A_{\max} - A_{\min}}{V_{\max} - V_{\min}}$$

V – Observed Variable
A – Presentable Variable

Once scaled training data set is prepared, it is ready for neural network training. Levenberg-Marquardt method (Martin T. Hagan and Mohammad B. Menhaj 1994) for solving the optimization is selected for back propagation training. It is selected due to its guaranteed convergence to a local minima, and its numerical robustness.

Experiments

The training data set is divided into two sets viz., dataset pairs which has Mach number less than 0.4, and those greater than 0.4². The data set is presented to the neural network architecture for the training. Initially a training set, which has 233 pairs, is presented to the neural network up to user-defined error of tolerance. The weights are stored and sparse data set of 9 pairs is provided for the same neural network architecture for further training. The initial training data set represents the exhaustive combination of data set in the particular parameter space. The initial training data set represents the general pattern of a particular aerodynamic coefficient. Based on the general pattern, the second training data set is interpolated. The initial data set is plotted in figure 1 and 2, and the data in figure 1 can be represented by a linear type of function whereas the data in figure 2 can be expressed as a combination of linear and hyperbolic tangent or sigmoid function. Numerous trials have been conducted with different combinations of transfer function be adopted for the input-to-hidden neurons and hyperbolic tangent or sigmoid function be used for the hidden-to-output layer. Figure 3 represents the sparse data set presented to the neural network successively after the initial training data set was presented. The figures 4 and 5 represent the neural network predicted data from the sparse data set. A few points are over fitted or under fitted in the results produced by the network. Over or under fitting is due to the sparseness of data. Overall the results produced by the network are considered to be very good.

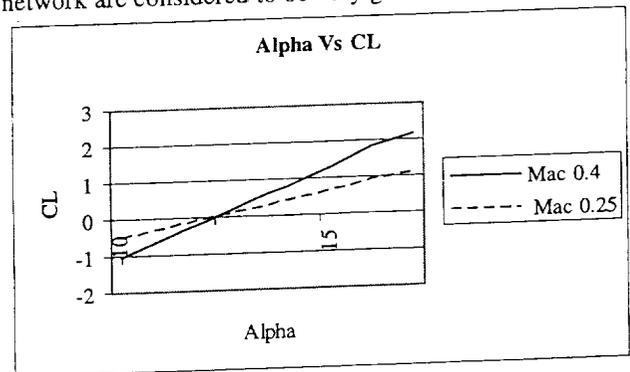


Figure 1 Initial Training data for neural network ($M \leq 0.4$)

² Mach number < 0.4 then data expressed as a linear function else a combination of linear and sigmoid function

Conclusion

Neural networks will become an important tool in future NASA Ames efforts to move directly from wind tunnel tests to virtual flight simulations. Many errors can be eliminated, and implementing a neural network can considerably reduce cost. Preliminary results have proved that neural network is an efficient tool to interpolate across sparse data. The prediction for the lower end and upper end of Mach number by the neural network is considerably deviated. The deviation is caused due to the non-availability of data in the sparse data. Initially neural network has been trained by the original data which enables network to understand overall pattern. Successive training by the sparse data alters the weights of the neural network which causes this deviation. This deviation is well within 10 %, which is acceptable in aerodynamic modeling.

Further research is focused to overcome this deviation in predicting sparse data. It is also directed to optimize number of hidden neurons and will be integrated into web-enabled application. Hybrid system using evolutionary theory and neural network is planned to build an efficient model to predict aerodynamic variables. The neural network will be an integral tool of data mining suite in an existing collaborative system in NASA.

Acknowledgements

We would like to acknowledge useful comments, suggestions and discussions from Dr. David Thompson and Ms. Joan Walton NASA Ames.

References

- Freeman J. A. and Skapura D. M. 1992. Neural networks and algorithms - Applications and Programming techniques. Addison-Wesley Publishing Company.
- Hagan M. T., Demuth H.B. and Beale M. H. 1996. Neural Network Design. Boston, MA. PWS Publishing, USA
- Jondarr. C. 1996. Backpropagation Family. Technical Report C/TR96-05. Macquarrie University.
- Kartalopoulos S.V. 1996. Understanding neural networks and fuzzy logic - basic concepts and applications. IEEE Press, USA.
- Lau C. 1992. Neural Networks theoretical foundations and analysis. IEEE Press.
- Martin T. Hagan and Mohammad B. Menhaj, 1994. Training Feed forward networks with the Marquardt Algorithm. IEEE Transactions on neural networks, Vol 5 No 6 989-993.
- Master T. 1993. Practical neural network recipes in C++. Academic Press Inc

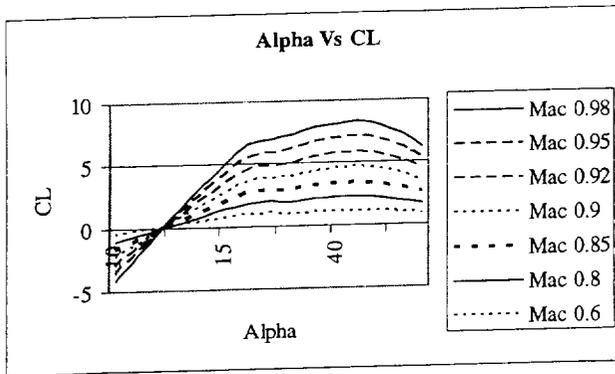


Figure 2 Initial Training data for neural network ($M > 0.4$)

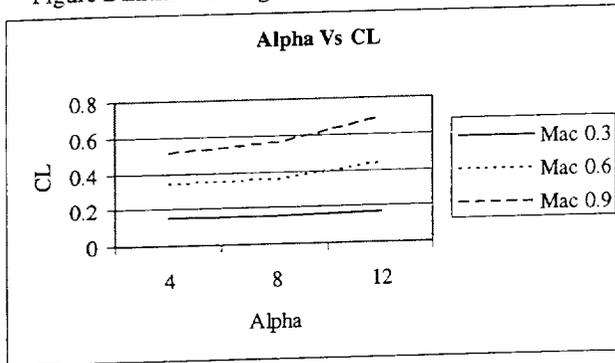


Figure 3. Sparse data presented to the neural network

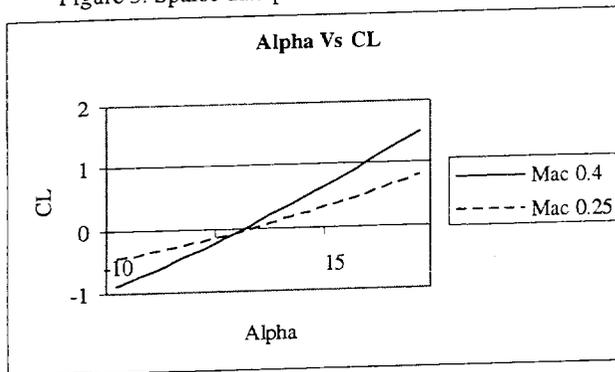


Figure 4 Neural network interpolated data for sparse data ($M \leq 0.4$)

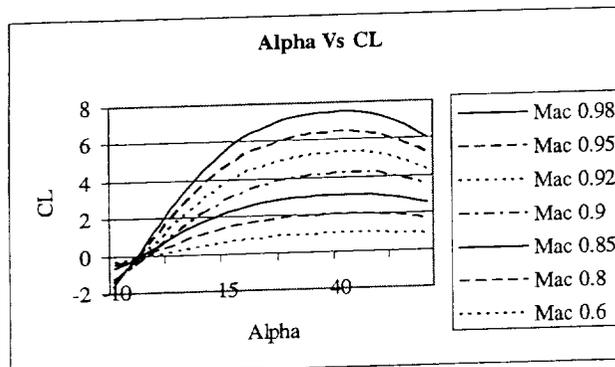


Figure 5 Neural network interpolated data for sparse data (Mach number > 0.4)

Rumelhart D.E., G.E.Hinton and R.J. Williams. 1986. "Learning internal representations by error propagation", in D.E. Rumelhart and J.L. McClelland, eds. *Parallel Data Processing*. Vol 1, Cambridge, MA: The MIT Press, 318-362.

Simpson P.K. 1990. *Artificial neural systems*, Pergamon Press.

Wasserman P. D. 1989. *Neural Computing - Theory and Practice*. Van Nostrand Reinhold Publishing

Ching F.Lo, J. L. Zhao and DeLoach R. 2000. Application of neural networks to wind tunnel data response surface methods. In proceedings of 21st AIAA aerodynamic measurement technology and ground testing conference, Denver, Colorado

Lawrence S, Lee Giles and Ah Chung Tsoi. 1996. What size neural network gives optimal generalization? Convergence properties of Backpropagation. Technical report UMIACS-TR-96-22 and CS-TR-3617

Lawrence S, Lee Giles and Ah Chung Tsoi 1997 Lessons in neural network training: overfitting may be harder than expected In proceedings of the fourteenth national conference on artificial intelligence, AAAI-97.

Magnus Norgaard, Jorgensen C and James Ross 1997, Neural network prediction of new aircraft design coefficients, NASA Technical memorandum 112197

Bruce Jackson and Christopher Cruz 1992, Preliminary subsonic aerodynamic model for simulation studies of the HL-20 lifting body, NASA technical memorandum 4302

M.M. Rai and N.K. Madavan 2000, Aerodynamic design using neural networks AIAA journal Vol 38. No 1pp 173-182

D.L. Elliott 1993 A better activation function for artificial neural networks ISR technical report TR93-8

Widrow B and Hoff. 1960. Adaptive swtiching circuits. In 1960 IRE WESCON Convention Record 96-104.

Widrow B and Stearns S. 1985. *Adaptive Signal Processing*. Prentice-Hall.

Widrow B., Winter and Baxter. 1987. Learning Phenomena in layered neural networks. In first Conference Neural Nets San Diego Vol 2 411